GraphTinker: Outlier Rejection and Inlier Injection for Pose Graph SLAM

GraphTinker: Outlier Rejection and Inlier Injection for Pose Graph SLAM

Linhai Xie[1], Sen Wang[2], Andrew Markham[1] and Niki Trigoni[1]

***Abstract*— In pose graph Simultaneous Localization and Mapping (SLAM) systems, incorrect loop closures can seriously hinder optimizers from converging to correct solutions, significantly degrading both localization accuracy and map consistency. Therefore, it is crucial to enhance their robustness in the presence of numerous false-positive loop closures. Existing approaches tend to fail when working with very unreliable front-end systems, where the majority of inferred loop closures are incorrect. In this paper, we propose a novel middle layer, seamlessly embedded between front and back ends, to boost the robustness of the whole SLAM system. The main contributions of this paper are two-fold: 1) the proposed middle layer offers a new mechanism to reliably detect and *remove* false-positive loop closures, even if they form the overwhelming majority; 2) artificial loop closures are automatically reconstructed and *injected* into pose graphs in the framework of an Extended Rauch-Tung-Striebel smoother, reinforcing reliable loop closures. The proposed algorithm alters the graph generated by the front-end and can then be optimized by any back-end system. Extensive experiments are conducted to demonstrate significantly improved accuracy and robustness compared with state-of-the-art methods and various back-ends, verifying the effectiveness of the proposed algorithm.**

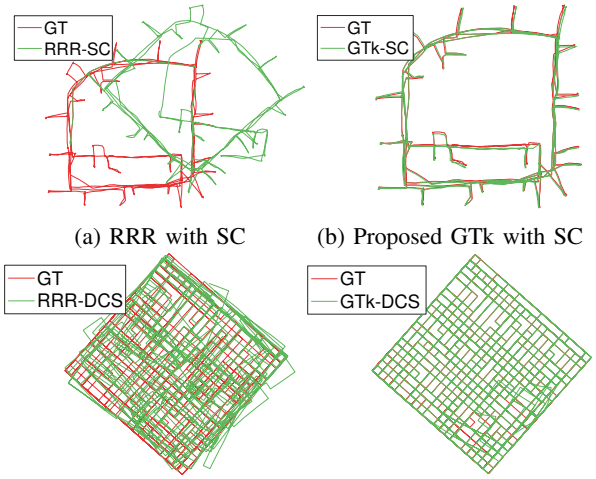
I. Introduction

Pose graph Simultaneous Localization and Mapping (SLAM), as one of the most popular and effective techniques for robot localization, has attracted significant interest over the past decade. A pose graph is built by sensor front-ends with edges indicating constraints (e.g. odometry observations, typically subject to long-term drift) and nodes representing robot poses or landmarks. Loop closures occur when revisiting an area, are represented by inserting additional edges or (constraints) into the graph. Pose graph SLAM back-ends, then, derive optimal robot locations by minimizing the errors of the pose graph.

In typical robotic applications, platforms are equipped with highly capable sensors e.g. LIDAR and vision, which can detect loop closures with a high degree of accuracy. However, in some emerging applications using less-informative sensors, e.g. the Earth's geomagnetic field for smartphone based localization, the number of ambiguous loop closures can be disproportionately higher than the true loop closures. Existing back-end optimizers are tolerant of a small proportion of incorrect loop closures, however, if the number of false-positive loop closures are excessively high (e.g. >50%) then the graph optimization can perform poorly or fail entirely.

Xie, Markham and Trigoni are with Department of Computer Science, University of Oxford, Oxford OX1 3QD, United Kingdom {firstname.lastname} @cs.ox.ac.uk

Wang is with School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh EH14 4AS, United Kingdom s.wang@hw.ac.uk



(a) RRR with SC (b) Proposed GTk with SC

(c) RRR with DCS (d) Proposed GTk with DCS

Fig. 1: RRR [8] and the proposed method GraphTinker (GTk) as a middle layer for pose graph SLAM. Red lines show ground truth (GT). (a)-(b) Results on Intel Research dataset with 1790 (200%) additional outliers based on Switchable Constraints (SC) [2] back-end. (c)-(d) Results on city10000 dataset with 21376 (200%) extra false-positive loop closures based on Dynamic Covariance Scaling (DCS) [3] back-end.

In recent years, significant effort has been made to improve the reliability of pose graph SLAM systems by enhancing the robustness of back-ends to false-positive loop closures. Existing algorithms [1]–[12] mostly focus on how to detect false-positive loop closures and further mitigate their impacts. Although these approaches have made significant strides, it is still very challenging for them to operate in the presence of numerous false-positive loop closures.

Since no algorithm can always remove all false-positive loop closures, the key to make pose graph SLAM systems more robust, in essence, is to greatly enlarge the proportion of inliers, namely correct loop closures, to outliers (false-positive loop closures) in pose graphs. To this end, we can not only *reject* false-positive loop closures, but also *inject* true-positive loop closures by reinforcing loop closures that are highly consistent with one another. In this paper, we demonstrate how this novel combination of approaches can yield significant benefits in the face of high proportions of erroneous loop closures like the example in Fig.1.

More specifically, our contributions are:

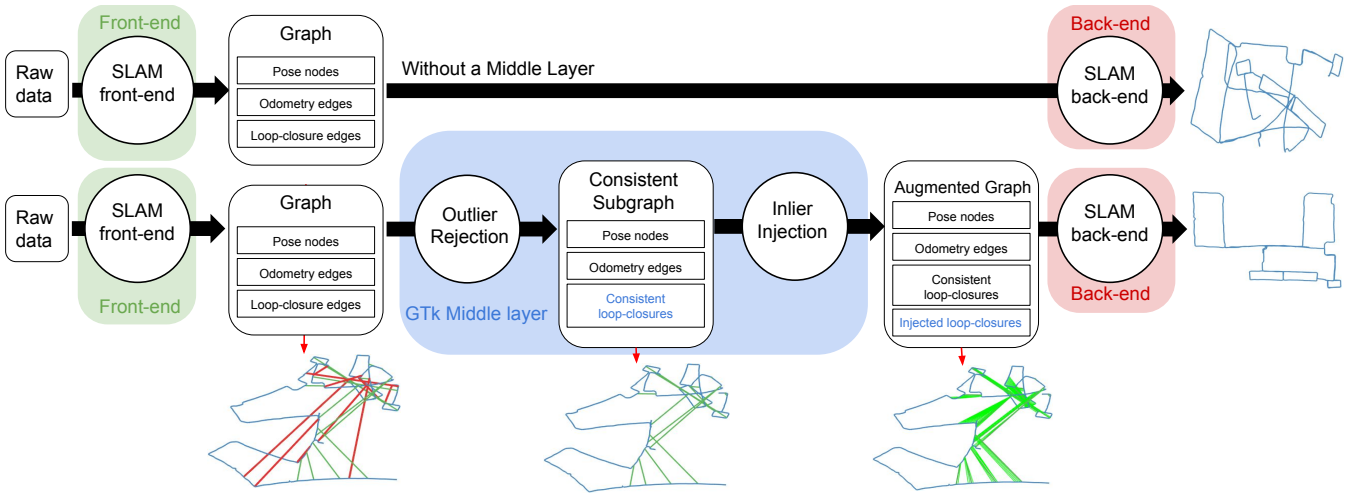


Fig. 2: System architectures without (top) and with (bottom) the proposed GTk middle layer. In GTk, inconsistent loop closures of a pose graph are first detected and eliminated by the outlier rejection module. Then, a set of artificial loop closures are reconstructed and inserted through the inlier injection module, generating an augmented graph for back-end. Three pose graphs at bottom give an example on the MIT-Killian-Court dataset. Red and light green lines represent false-positive and injected loop closures, respectively.

- A spatial consistency testing algorithm is proposed to measure the consistency between a pair of loop closures and select a subset of consistent ones, removing false-positive loop closures even if they form an overwhelming majority.
- A novel approach is developed to boost the robustness of pose graph SLAM by automatically reconstructing and injecting trustworthy artificial loop closures in the framework of an Extended Rauch-Tung-Striebel smoother.
- We construct a universal middle layer, named Graph-Tinker(GTk), from the above two approaches, which can be used in conjunction with any front-end and back-end systems to further improve their performance.
- We demonstrate the performance of GTk in comparison with the state-of-the-art technique RRR [8], showing great improvements in graph reconstruction.

Please note that the proposed GTk as a middle layer is a complement to back-ends, operating in tandem with them.

We first discuss related work in Sec. II, followed by the overview of the proposed system in Sec. III. The key technique to reject and reconstruct loop closures is detailed in Sec. IV. Experimental results are given in Sec. V before drawing conclusions in Sec. VI.

II. Related Work

Over the last few years, several robust back-end algorithms have been proposed to tackle problems with accuracy caused by false-positive loop closures. They can be roughly divided into two types: approaches based on augmented models, and approaches based on graph consistency.

A. Approaches based on Augmented Models

Augmented model based methods focus on how to model pose graph SLAM problem by taking false-positive loop closures into consideration and reject outliers during graph optimization.

Sünderhauf and Protzel [1], [2] indicate that the topology of a factor graph can be partially unfixed and present the idea of Switchable Constraints (SC) where a switchable variable is given to each loop closure constraint. A constraint is turned off during optimization once it is considered as an outlier. Based on this work, Agarwal et al. [3] introduce Dynamic Covariance Scaling (DCS) which replaces the quadratic cost metric with an m-estimator and reaches a faster convergence.

Olson and Agarwal [5], [6] create a mixture model by merging two different Gaussian models. Their main insight is to use a max operator between models rather than a sum operator as it can largely simplify the solution of posterior maximum likelihood.

Lee et al. [7] model the robust back-end problem as a Bayesian network and apply a Classification Expectation Maximization algorithm to solve it. An additional variable is assigned as the weight of each loop closure constraint and finally the weights of outliers are decreased to mitigate the influence of outliers during optimization.

Although the augmented model based method can achieve superior performance with a reasonable number of false-positive loop closures, there are some drawbacks. For example, as reported in [3], DCS tends to be less effective for randomly distributed outliers. Furthermore, they all heavily rely on parameter tuning.

B. Approaches based on Graph Consistency

Graph consistency based approaches aim at selecting a subset of loop closures for pose optimization based on consistency checks. The Realizing, Reversing, Recovering (RRR) algorithm [4], [8] first divides all the loop closures into several clusters according to timestamps and then applies a number of χ^2 tests to check both intra- and inter- cluster

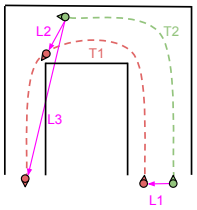


Fig. 3: Example showing loop closures. Loop closures L_1 and L_2 are true-positive, while L_3 is a false-positive one arising from sensor noise in the front-end. L_1 and L_2 reflect correct relative poses between their terminal nodes while L_3 does not.

consistency. Individual loop closures in a cluster, or an entire cluster, with high χ^2 values are rejected.

More recently [9] and [10] also focus on graph consistency. However, unlike RRR, they search for the largest consistent subset of the constraints.

This type of approach is most related to the proposed GTk method since they work in a middle-layer and complement back-end optimization approaches. However, the consistency test proposed in [8] requires classification of loop closures into different clusters (involving a parameter to decide the number of clusters) and then measures the consistency within or between clusters. In contrast, the proposed method checks consistency in the context of a *pair* of loop closures, and thus does not require any knowledge about the underlying distribution.

There are some further works which do not belong in the above two categories. Segal et al. [11] proposes a new optimization approach which applies hybrid inference on the Bayes tree. It combines non-linear least squares with discrete inference and use discrete states to enable or disable measurements. Fourie et al. [12] propose a non-parametric method and obtain a more general solution to the Bayes Tree.

III. System Overview

The architectures of two systems with and without the GTk middle layer are shown in Fig. 2. The proposed middle layer lies between the front and back ends with pose graphs as input and output. Therefore, the proposed method can be directly used in conjunction with any kind of front and back end generating and optimizing pose graphs, respectively.

A. Front-End

Since we are building a universal middle layer that only needs a pose graph from the front-end, any front-end system that produces a pose graph should be compatible with the proposed GTk middle layer.

B. Middle Layer

There are two main steps in the proposed GTk middle layer. After a noisy pose graph is built by the front-end, the GTk middle layer first detects and removes false-positive loop closures through the outlier rejection module, retaining consistent loop closures. A subset of these is then selected to reconstruct artificial loop closures and create an augmented graph based on the inlier injection module. Recovering

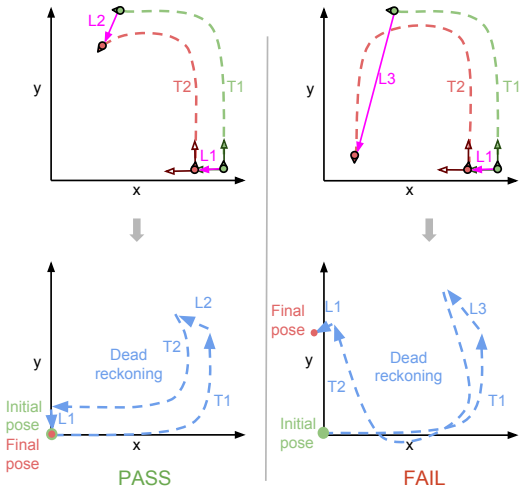


Fig. 4: Spatial consistency check of a pair of loop closures. Left: the pair of loop closures L_1 and L_2 passes the spatial consistency test since they almost return back to the initial pose after traversing the circle. Right: loop closures L_1 and L_3 are inconsistent with one another due to the incorrect relative pose in L_3.

correct but undetected loop closures is of importance since existing loop closure detection algorithms cannot ensure a 100% recall with high precision.

C. Back-End

Given the augmented pose graph, the SLAM back-end can calculate optimal robot poses through optimization. For our experiments, we utilize g2o [13] as it is one of the most popular back-end frameworks for graph SLAM. Please refer to [14] and [15] for more details on graph SLAM and its back-end.

IV. GTk Middle Layer for Loop Closure Selection and Reconstruction

The GTk middle layer aims at solving two problems. One is to search for reliable loop closures and filter out false-positive ones. The other is to reinforce correct loop closures by establishing additional loop closures in their vicinity.

Fig. 3 illustrates an example where a robot passes through same place twice with trajectory segments T_1 and T_2, respectively. Assume that the front end discovers three loop closures - L_1, L_2 and L_3 that link nodes in the two trajectories. A loop closure typically contains information about the connectivity of two nodes, relative pose between them and corresponding covariance. Suppose only loop closures L_1 and L_2 are true-positives, i.e., their relative poses are correct, whereas loop closure L_3 is a false-positive, i.e. one having an incorrect relative transformation. In the next subsection, we explain how to apply spatial consistency checks to estimate whether a loop closure is correct or not.

A. Detection and Selection of Consistent Loop Closures

1) Spatial Consistency Test for a Pair of Loop Closures: The key idea is to apply a spatial consistency test to each pair

of loop closures in the constructed graph. Fig. 4 demonstrates this for two example pairs, namely (L_1,L_2) and (L_1, L_3). The intuition is that the relative poses encapsulated in the loop closure pair should be consistent with the odometry information of the outbound and inbound trajectory segments. Two loop closures and the trajectories between them form a closed chain, or circle in the graph, as shown in Fig. 4. The key assumption that we make is that odometry edges only suffer from limited cumulative drift, and don't experience large abrupt odometry errors. Under this assumption, the consistency of the loop closure pair can be tested based on the combination of a probability propagation by dead-reckoning and a χ^2 test.

More specifically, we draw inspiration from the statistics method in [16] to measure the consistency between two loop closures. For simplicity, let us start at the initial point of trajectory T_1 with an initial pose vector and a zero covariance matrix. With the relative pose information provided by both odometry edges and loop closure edges, the probability distribution of final pose after traversing the circle can be calculated through dead reckoning. Note that we need to recalculate the relative pose and covariance matrix of the odometry edges in the inbound trajectory T_2 and loop closure L_1 in a reverse direction to form a unidirectional circle. Then, we can assume that the mean of the distribution of the final pose should be close to the initial pose if these two loop closures are highly consistent with each other. This is because the error provided by odometry drift should be small in comparison to the length of the loop. Thus, we have our null hypothesis of a distribution where the mean is a zero vector (the same as initial pose) and the covariance is equal to the calculated final pose. Lastly, a χ^2 test is applied to check whether the final pose is accepted as the null hypothesis.

As shown in Fig. 4, since the loop closures L_1 and L_2 are correct and their relative poses recorded by the front-end system is correct, the probability of returning back to the near neighbor of the initial pose after traversing the outbound and inbound trajectories is relatively high. In contrast, this is not applicable to L_1 and L_3 due to the erroneous relative pose between two nodes from the false-positive loop closure L_3.

Based on the pose nodes, odometry edges and loop closure edges, it is straightforward to calculate the probability distribution of a final pose matching with its initial pose. Let $\mathbf{p}_0^{T_i}$, $\mathbf{p}_1^{T_i}$, ..., $\mathbf{p}_{n_i}^{T_i}$ be the poses of trajectory T_i, and $\mathscr{N}(\mathbf{u}_1^{T_i}, \mathbf{S}_1^{T_i})$, ..., $\mathscr{N}(\mathbf{u}_{n_i}^{T_i}, \mathbf{S}_{n_i}^{T_i})$ the normal distributions of the corresponding odometry edges. Taking loop closures L_1 and L_2 as an example, let the two loop closure edges follow distributions $\mathscr{N}(\mathbf{l}_1, \mathbf{S}_1^l)$ and $\mathscr{N}(\mathbf{l}_2, \mathbf{S}_2^l)$, respectively. Note that the above data are all available directly in the constructed pose graph. The relative poses and covariance matrices in the inbound trajectory segment $T2$ and loop closure $L1$ need to be reversed to form the unidirectional circle. Define the probability distribution $\mathbf{p}_0$ of the initial pose as

$$\mathbf{p}_0 \sim \mathscr{N}(\mathbf{m}_0, \mathbf{S}_0), \quad \mathbf{m}_0 = (x_0, y_0, \theta_0)^T, \quad \mathbf{S}_0 = \mathbf{0} \tag{1}$$

where $\mathbf{0}$ is a 3×3 zero matrix. Therefore, the distribution of the final pose $\mathbf{p}_{fin} \sim \mathscr{N}(\mathbf{m}_{fin}, \mathbf{S}_{fin})$ can be derived with the following chaining equations:

$$\begin{aligned}
&\mathbf{m}_k = f(\mathbf{m}_{k-1}, \mathbf{i}_k), k \in 1, 2, ..., n_1 + n_2 + 2, \\
&\mathbf{i}_k \in \{\mathbf{u}_1^{T_1}, ..., \mathbf{u}_{n_1}^{T_1}, \mathbf{l}_1, \mathbf{u}_1^{T_2}, ..., \mathbf{u}_{n_2}^{T_2}, \mathbf{l}_2\}, \\
&\mathbf{S}_k = \mathbf{S}_{k-1} + \mathbf{J}_k \mathbf{S}_k \mathbf{J}_k^T + \mathbf{V}_k, \\
&\mathbf{J}_k = \frac{\partial f}{\partial \mathbf{i}_k}, \quad \mathbf{V}_k \in \{\mathbf{S}_1^{T_1}, ..., \mathbf{S}_{n_1}^{T_1}, \mathbf{S}_1^l, \mathbf{S}_1^{T_2}, ..., \mathbf{S}_{n_2}^{T_2}, \mathbf{S}_2^l\}
\end{aligned} \tag{2}$$

where $\mathbf{m}_k$ and $\mathbf{S}_k$ are the mean and covariance of the dead reckoning at time k, $\mathbf{i}_k$ and $\mathbf{V}_k$ are mean and covariance of the control variable at time k, and $f(\cdot)$ is robot's motion model. With $\mathbf{p}_{fin} = \mathbf{p}_{n_1+n_2+2}$ indicating the probability distribution of the final pose, a χ^2 test is applied to decide whether the calculated distribution should be accepted as the null hypothesis where the final pose is a zero mean distribution. If $\mathbf{m}_{fin}^T \mathbf{S}_{fin} \mathbf{m}_{fin} < \chi^2(\alpha)$ we accept the null hypothesis and consider that this pair of loop closures is consistent with a confidence of $1 - \alpha$. $\chi^2(\alpha)$ is the inverse function of chi-square cdf and α is set to be 0.1 as default.

Measuring the spatial consistency on each pair of loop closures is undesirable since long trajectory segments suffer from large accumulative drifts on odometry, leading to an inaccurate result of spatial consistency test. Another reason is that computational overhead grows quadratically w.r.t the number of loop closures. Thus, in our implementation we set a limitation on the maximum distance of two trajectory segments between loop closures.

2) Loop Closure Selection: When all spatial consistency tests are finished, each loop closure L may pass n_p tests and fail in n_f tests. Now we can define a ratio between n_p and $n_p + n_f$ which is named as pass rate.

After calculating the pass rate of each loop closure, we use k-means clustering to classify all loop closure into two groups according to the pass rate. Based on k-means, each group will have a centre value. Therefore the one with a higher centre value of pass rate will be regarded as the group containing true positives and is called the inlier group while the other one is call outlier group. Then, the loop closures in the outlier group is rejected as false-positives. In the inlier group, loop closures with the pass rate greater than the group centre value is further classified into a class of strongly consistent loop closures used for artificial loop closure construction.

Therefore, after the above loop closure detection and selection, a consistent subgraph where, at least, the majority of the loop closures are true-positive is produced. It is worth noting that it is not necessary to eliminate out all false-positives at this stage because in the subsequent reconstruction steps, artificial loop closures will be constructed and injected to further reinforce true-positive loop closures of this subgraph.

B. Loop Closure Reconstruction

In this section, we explain how artificial loop closures are reconstructed and then introduced into a subgraph previously generated with selected consistent loop closures.

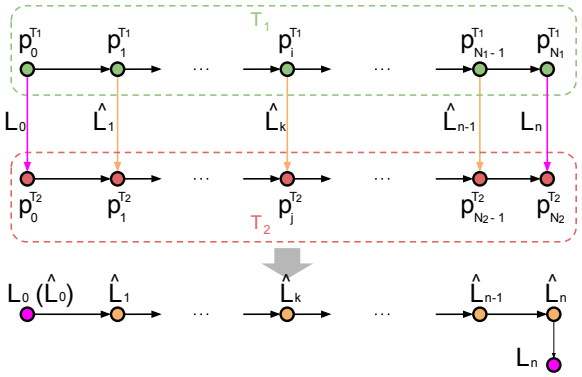


Fig. 5: Formulating graphical model for all potential artificial loop closures. $\hat{L}_0, \hat{L}_1, ..., \hat{L}_n$ are potential loop closures from the two trajectory segments T_1 and T_2. L_0 is directly regarded as the initial artificial loop closure $\hat{L}_0$ and L_n is modeled as the observation of the last artificial loop closure $\hat{L}_n$.

1) Searching Neighbor: Each consistent loop closure is first assigned with a consistent neighbor loop closure, which is discovered by a shortest path algorithm among loop closures. More specifically, by starting at one terminal of the selected loop closure, a Dijkstra's shortest path algorithm is employed to find another loop closure that is on the shortest path between two terminals of the selected loop closure and is consistent with it. The search result is accepted if and only if a unique loop closure exists on this shortest path and the length of the shortest path is under a maximum distance threshold.

The reason for only constructing artificial loop closures between a loop closure and its consistent neighbor rather than all consistent loop closures is that a spatial consistency test is more reliable when the distance traversed on the circle is shorter due to odometry drifts.

2) Calculating Relative Pose between Trajectory Segments: For each selected loop closure that has a strongly consistent neighbor, artificial loop closures are constructed on the two trajectory segments between them. To this end, two kinds of information are required. One is connectivity indicating which two pose nodes should be connected by an artificial edge. The other is a relative transformation between the two pose nodes and its covariance matrix. To calculate them correctly, knowing the relative transformation between two trajectory segments properly is of importance.

Since the relative transformation between the two trajectory segments can be defined by either the selected loop closure or its consistent neighbor loop closure, the one having passed more spatial consistency tests, which indicates a stronger consistency, is used. If the two loop closures coincidentally have equal consistencies, one is randomly chosen. Then, one of the trajectory segments can be transformed to have a roughly correct relative pose w.r.t another trajectory segment according to the relative transformation of the two poses of the chosen loop closure.

3) Establishing Connectivity of Potential Loop Closures: Once the relative transformation between two trajectory segments are known and one trajectory segment is rotated to have correct relative pose w.r.t the other, the next step is to construct connectivity of artificial loop closures. Dynamic Time Warping (DTW) [17] is utilized to match poses of the two trajectories. Since DTW only allows associating trajectories traversed in the same direction, one trajectory segment would be reversed if the directions of two trajectory segments are different. All matches of poses between the two trajectory segments are the potential artificial loop closures, which, however, only contain connectivity without transformation and covariance information needed for a loop closure. The subsequent subsections focus on how to compute this information.

4) Formulating Model for Potential Artificial Loop Closures: To calculate relative transformation, including translation and rotation, and corresponding covariance of each potential artificial loop closure, we formulate a graphical model as in Fig. 5. In the upper part, there are two trajectory segments T_1 and T_2 which contain pose nodes (green and red nodes) and odometry edges (black arrows). The selected loop closure and its consistent neighbor are L_0 and L_n respectively (magenta arrows). Since the connectivity of all the potential artificial loop closures are established in the last step, they are represented by $\hat{L}_0, \hat{L}_1, ..., \hat{L}_n$ (yellow arrows) in the figure.

The basic idea is that once the odometry information of the trajectory segments between any two loop closures is available, the distribution of one loop closure can be deduced from the distribution of the other one through probability propagation by dead reckoning. Therefore, by considering the selected loop closure L_0 as the initial loop closure and its consistent neighbor L_n as the observation of the last loop closure $\hat{L}_n$, we can obtain a loop closure chain as shown in the bottom part of the figure. All loop closures, including artificial ones and two real ones, are modeled into nodes (magenta and yellow nodes) in the model whose relative pose and covariance will be calculated next.

5) Calculating Relative Pose and Covariance: The distribution of all the artificial loop closures can be calculated by dead reckoning where each loop closure rather than a pose is taken as the state variable. The probability is propagated from the initial loop closure through an Extended Rauch-Tung-Striebel (ERTS) smoother and the propagated mean and variance are constrained by the observation of the last artificial loop closures to avoid divergence. Note that all the artificial loop closure mentioned above are still potential ones.

Assume $\hat{\mathbf{L}}_k \sim \mathcal{N}(\hat{\mathbf{x}}_k, \hat{\mathbf{S}}_k^l), (k = 0, 1, 2, ..., n)$ are potential artificial loop closures where $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{S}}_k^l$ are the state variable and covariance we intend to calculate, except for the initial one which is the same as the selected loop closure, i.e., $\hat{\mathbf{L}}_0 = \mathbf{L}_0 \sim \mathcal{N}(\mathbf{l}_0, \mathbf{S}_0^l)$. While $\mathbf{L}_n \sim \mathcal{N}(\mathbf{l}_n, \mathbf{S}_n^l)$, which actually is the consistent neighbor, is regarded as an observation of the last potential artificial loop closure $\hat{L}_n$ in the model. Note that all $\hat{\mathbf{x}}_k$, $\mathbf{x}_0$ and $\mathbf{x}_n$ are vectors indicating relative poses in the loop closures while $\hat{\mathbf{S}}_k^l$, $\mathbf{S}_0^l$ and $\mathbf{S}_n^l$ are covariance matrices. Furthermore, the odometry edges of trajectory segments T_1

and T_2 between two potential artificial loop closures are represented by $\mathbf{o}_i^{T_1} \sim \mathcal{N}(\mathbf{u}_i^{T_1}, \mathbf{Q}_i^{T_1}), (i = 1,2,...,n_1)$ and $\mathbf{o}_i^{T_2} \sim \mathcal{N}(\mathbf{u}_j^{T_2}, \mathbf{Q}_j^{T_2}), (j = 1,2,...,n_2)$ respectively. Due to the DTW algorithm utilized in our implementation, there will be no more than one odometry edge from each trajectory segment between one potential artificial loop closure and its successor. Because of the limited length of paper, we only describe the case with two odometry edges between two loop closures which is more complicated than the other cases.

Since a standard ERTS smoother mentioned in [18] is applied, we only address the transition model and the observation model used in the smoother. The former solves the propagation from a potential artificial loop closure $\hat{\mathbf{L}}_{k-1}$ to the successor $\hat{\mathbf{L}}_k$ with odometry edges $\mathbf{o}_i^{T_1}$ and $\mathbf{o}_j^{T_2}$ as follows:

$$\begin{aligned}\hat{\mathbf{x}}_k &= g(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_j^{T_2}, \mathbf{u}_i^{T_1}) \\ &= \mathbf{M}(\Delta\theta_i^{T_1})[f(f(\mathbf{0}, \hat{\mathbf{x}}_{k-1}), \mathbf{u}_j^{T_2}) - f(\mathbf{0}, \mathbf{u}_i^{T_1})]\end{aligned} \tag{3a}$$

$$\hat{\mathbf{S}}_k^l = \mathbf{J}_{\hat{\mathbf{x}}_{k-1}} \mathbf{S}_{k-1}^l \mathbf{J}_{\hat{\mathbf{x}}_{k-1}}^T + \mathbf{J}_{\mathbf{u}_j^{T_2}} \mathbf{S}_j^{T_2} \mathbf{J}_{\mathbf{u}_j^{T_2}}^T + \mathbf{J}_{\mathbf{u}_i^{T_1}} \mathbf{S}_i^{T_1} \mathbf{J}_{\mathbf{u}_i^{T_1}}^T \tag{3b}$$

$$\mathbf{J}_{\hat{\mathbf{x}}_{k-1}} = \frac{\partial g}{\partial \hat{\mathbf{x}}_{k-1}}, \quad \mathbf{J}_{\mathbf{u}_j^{T_2}} = \frac{\partial g}{\partial \mathbf{u}_j^{T_2}}, \quad \mathbf{J}_{\mathbf{u}_i^{T_1}} = \frac{\partial g}{\partial \mathbf{u}_i^{T_1}} \tag{3c}$$

where (3a) is the transition from $\hat{\mathbf{L}}_{k-1}$ to $\hat{\mathbf{L}}_k$. Its non-linearity is the reason why we choose ERTS rather than RTS smoother. Matrix $\mathbf{M}$ is a rotation matrix and $\Delta\theta_i^{T_1}$ is orientation of $\mathbf{u}_i^{T_1}$, representing the relative rotation in this odometry edge. $f()$ is the robot motion model. And $\mathbf{0}$ is a zero vector. (3b) shows how to propagate the covariance matrix where $\mathbf{J}$ denotes the Jacobian matrix of function $g()$ w.r.t to each input vector.

The above equations propagate distributions in the forward process when there is no observation of the potential artificial loop closure. For the last loop closure $\hat{\mathbf{L}}_n$ which has an observation $\mathbf{L}_n$, we use the following observation model to execute an update step.

$$\hat{\mathbf{y}}_n = \mathbf{H}\hat{\mathbf{x}}_n + \mathbf{r}_n, \quad \mathbf{r}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{S}_n^l) \tag{4}$$

where matrix $\mathbf{H}$ is a 3×3 identity matrix, it sums up the mean value of the last potential artificial loop closure $\hat{L}_n$ with a zero-mean Gaussian noise whose covariance function is the same as the one of its neighbor, the consistent neighbor loop closure L_n. Due to the linearity, the calculation of partial derivation is avoided for the observation model.

6) Choosing Potential Loop Closures to Inject: Among all potential artificial loop closures, only 10% of them that have the smallest uncertainties are injected into the graph, producing an augmented pose graph for back-ends.

V. Experimental Results

The proposed GTk algorithm is implemented in Matlab and tested on eight public datasets. In terms of competing approach, we consider RRR [8] as another middle layer, and back-ends (DCS [3], SC [1] and Cauchy robust kernel [19]), and directly use the open-source implementations.

TABLE I: Public datasets used in experiments

Dataset	Poses	Original loops	Max additional outliers
Bicocca	43116	767	1534
Bovisa04	11393	197	394
Bovisa06	10744	219	483
city10000	10000	10688	21376
M3500Olson	3500	2099	4198
ringCity	2361	901	1802
Intel	943	895	1790
MIT	808	20	40

A. Datasets

To fairly assess our approach and compare it with others, we use 8 different public datasets for experiments. Bicocca, Bovisa04 and Bovisa06 datasets are from [8]. Manhattan3500Olson (M3500Olson), ringCity, city10000 and intel datasets are available in the open source package of vertigo [2] (`https://openslam.org/vertigo`). MIT-Killian-Court(MIT) dataset is also from open source (`http://www.lucacarlone.com/index.php/resources/datasets`).

In each dataset, varying numbers of additional outliers are randomly generated according to the number of original loop closures in the graph (25%, 50%, 100%, 200%). Thus for each dataset, we create four additional sets. Note that high proportions of false positive loop closures truly exist under some circumstances, e.g. when applying geo-magnetic information for loop closure detection as in [20] and the threshold is not tuned well, or when utilizing [21] where the testing data is largely different from training data. The relative pose in each outlier is sampled from a uniform distribution in Special Euclidean Group SE(2) while the information matrix is set to the average value of information matrices of original loop closures in a graph.

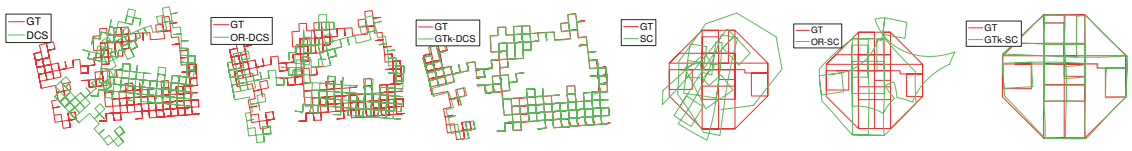
B. With and Without GTk Middle Layer

In this section, we validate the enhanced robustness of the whole system when only the outlier rejection (OR) algorithm or the entire GTk algorithm is employed as a middle layer between the front and back-ends. This gives insight into the relative contribution of each component. Three robust back-end algorithms, Cauchy robust kernel (Cauchy) [19], DCS [3] and SC [1], implemented in g2o and vertigo are adopted to use in conjunction with the proposed middle layer. We compare the performance of these back-ends when they are combined with and without OR/GTk on all datasets with a growing number of outliers. Although we have done experiments on all datasets with several numbers of outliers, due to limited space, only results from M3500Olson, ringCity and MIT datasets with 50%, 100% and 200% outliers are illustrated in Table II and Fig. 6.

In Table II, the root-mean-square error (RMSE) of optimized pose graphs are shown. As we can see, when using the proposed outlier rejection(OR) algorithm alone, the system becomes much more robust to false positive loop closures. And when GTk is applied, the robustness of the system is further enhanced, generally increasing accuracy. Although there are few cases in which the error actually increases

TABLE II: Results of different back-ends combined with no middle layer, Outlier Rejection (OR) only or GTk (outlier rejection and inlier insertion) on 4 datasets

Dataset		ringCity			M3500Olson			Intel			MIT-Killian-Court		
Back-end		Cauchy	DCS	SC	Cauchy	DCS	SC	Cauchy	DCS	SC	Cauchy	DCS	SC
Outliers	Middle layer	RMSE(m)											
50%	None	10.88	0.72	**0.66**	23.87	9.11	0.27	**0.04**	0.06	0.12	12.61	16.37	13.71
	OR	0.95	**0.48**	1.12	1.98	0.23	0.37	0.05	0.06	0.06	**1.87**	9.78	**1.84**
	GTk	**0.68**	0.83	1.12	**0.19**	**0.09**	**0.09**	0.07	**0.06**	**0.06**	1.90	**2.28**	2.09
100%	None	31.03	4.78	5.23	24.60	8.15	**0.01**	0.05	0.06	0.08	16.87	52.51	19.99
	OR	2.25	**0.66**	0.70	16.87	0.12	0.13	**0.05**	**0.06**	0.06	2.32	13.61	8.25
	GTk	**1.82**	0.68	**0.63**	**6.68**	**0.07**	0.08	0.07	0.06	**0.06**	**1.27**	**10.61**	**8.25**
200%	None	51.63	13.42	23.42	25.43	8.03	**0.02**	0.09	**0.06**	0.20	53.18	58.22	127.14
	OR	3.74	**0.64**	22.21	23.48	2.75	0.25	**0.06**	0.06	0.06	**36.22**	72.46	64.14
	GTk	**1.44**	1.08	**0.91**	**16.35**	**0.20**	0.20	0.07	0.06	**0.06**	36.25	**26.04**	**64.11**



(a) (b) (c) (d) (e) (f)

Fig. 6: Some typical results of different back-ends on two datasets with 200% additional outliers. We show the impact of no middle layer, using only our outlier rejection (OR) approach or the entire GraphTinker (GTk) system. The red trace shows the ground truth. (a) - (c) Results of DCS, OR-DCS and GTk-DCS respectively on Manhattan3500Olson dataset, using DCS. (d) - (f) SC without any middle layer, with OR, and with GTk on ringCity dataset.

slightly when GTk is applied, we mostly observe dramatic improvements in the robustness of the whole system, enabling back-ends to converge to correct results.

The specific influence brought by different parts of middle layers is more clearly demonstrated in Fig.6 with the Mahattan3500Olson and ringCity datasets. It can be seen that the optimization result is largely improved by our outlier rejection(OR) algorithm because it effectively removes most of the false-positive loop closures. However, since a number of true-positives are also rejected, the graph loses some essential constraints, and does not converge globally to the ground-truth. But when the entire GTk is applied, these discarded constraints are artificially constructed by our inlier injection algorithm which enables the back-end to converge to an accurate reconstruction. In general, it can be seen that the combination of outlier rejection and inlier insertion together can make existing SLAM approaches significantly more robust, without extensive parameter tuning.

C. Comparison with RRR

For existing robust graph SLAM algorithms and back-ends, RRR is the most similar to the proposed GTk with open-source resources. Therefore, it is chosen as a comparison. The combinations of RRR and GTk as a middle layer with DCS and SC back-ends are tested with all datasets. Some of the results are shown in Fig. 7 and Fig. 8. It is worth noting that we keep using the default parameters values in both RRR and GTk in all the experiments as we believe a robust middle layer should not rely on manually tuned parameters.

Fig. 7 illustrates the RMSE of the results. Each subgraph represents the experiments on a dataset where the x axis indicates the ratio between the number of additional false-positives and original loop closures. In Fig. 7a, although GTk tends to have a worse performance when combined with SC, the RMSE is extensively reduced when DCS is applied as the robust back-end on the Bovisa04 dataset. In all these cases, GTk outperforms RRR, especially on the M3500Olson and ringCity datasets. For more detailed results and comparison with RRR, please refer to the supplementary file.

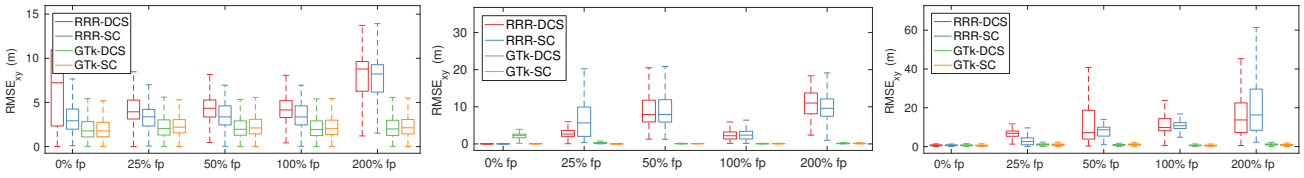
TABLE III: Runtime of GTk on 8 datasets.

Dataset	Number of loop closures	Running time (s)
Bicocca	767	14.18
Bovisa04	197	3.22
Bovisa06	219	7.34
city10000	10688	809.69
M3500Olson	2099	315.06
ringCity	901	88.18
Intel	895	311.39
MIT	20	0.25

D. Runtime Analysis

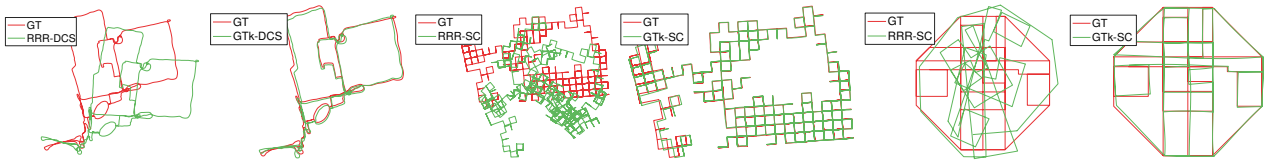
Although we only implement GTk with Matlab rather than C or C++, the runtime of GTk is still reasonable. The detailed runtime of GTk on the eight datasets is given in Table. III. Furthermore, by continuously increasing the number of loop closures in Manhattan3500Olson dataset, we find that the runtime of our algorithm has a linear growth w.r.t the increasing number of loop closures.

The most time consuming part in GTk is executing spatial consistency tests while searching for a consistent subset of loop closures in the graph. We avoid the quadratic time



(a) Bovisa04 (b) M3500Olson (c) ringCity

Fig. 7: Comparison between applying RRR and GTk as the middle layer on several datasets, combined with two different robust back-ends (DCS and SC). In each dataset, a growing percentage of additional random false-positives are generated. Red and blue boxes (first two in each cluster) represent results from RRR while green and orange boxes (last two in each cluster) are for GTk.



(a) (b) (c) (d) (e) (f)

Fig. 8: Some results on three datasets for the comparison of utilizing RRR or GTk as middle layer, combined with DCS and SC as robust back-ends. Red line is the ground truth. (a) - (b) RRR and GTk results on Bovisa04 datasets with 200% outliers when DCS as back-end. (c) - (d) RRR and GTk results on M3500Olson dataset with 200% extra false-positives when SC as back-end. (e) - (f) RRR and GTk results on ringCity dataset with 200% additional outliers with SC as back-end.

increase w.r.t the number of loop closures by restricting the traversal distance of the circle in the test with a threshold which significantly reduces the number of loop closures that need to be compared. Hence, this threshold determines runtime to some extent. The default value is set to 300 steps to achieve a suitable balance between runtime and performance.

VI. Conclusions and Future Work

This paper proposes a novel algorithm for boosting the robustness of a pose graph SLAM system to false-positive loop closures. We suggest that this robustness could be achieved by not only removing unreliable loop closures but also reconstructing additional loop closures in the graph. Extensive experiments verify that the proposed algorithm can successfully handle very noisy pose graphs where the majority of loop closures are false-positives, achieving more accurate results than back-ends only and competing algorithms. As future work, we will implement GTk with C++ to speed it up and will explore an extension for 3D datasets.

Acknowledgment

The authors would like to thank Ronnie Clark for the helpful suggestions on writing this paper.

References

[1] N. Sünderhauf and P. Protzel, "Towards a robust back-end for pose graph slam," in *ICRA*. IEEE, 2012, pp. 1254–1261.
[2] N. Sunderhauf and P. Protzel, "Switchable constraints for robust pose graph slam," in *IROS*. IEEE, 2012, pp. 1879–1884.
[3] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *ICRA*. IEEE, 2013, pp. 62–69.
[4] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time," in *Proc. Robotics: Science Systems*, 2013, pp. 233–240.
[5] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 826–840, 2013.
[6] E. Olson, "Inference on networks of mixtures for robust robot mapping," *Robotics: Science and Systems VIII*, p. 313, 2013.
[7] G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Robust pose-graph loop-closures with expectation-maximization," in *IROS*. IEEE, 2013, pp. 556–563.
[8] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time for pose graph slam," *The International Journal of Robotics Research*, pp. 1611–1626, 2013.
[9] L. Carlone, A. Censi, and F. Dellaert, "Selecting good measurements via l-1 relaxation: A convex approach for robust estimation over graphs," in *IROS*. IEEE, 2014, pp. 2667–2674.
[10] M. C. Graham, J. P. How, and D. E. Gustafson, "Robust incremental slam with consistency-checking," in *IROS*. IEEE, 2015, pp. 117–124.
[11] A. V. Segal and I. D. Reid, "Hybrid inference optimization for robust pose graph estimation," in *IROS*. IEEE, 2014, pp. 2675–2682.
[12] D. Fourie, J. Leonard, and M. Kaess, "A nonparametric belief solution to the bayes tree," in *IROS*. IEEE, 2016, pp. 2189–2196.
[13] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *ICRA*. IEEE, 2011, pp. 3607–3613.
[14] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.
[15] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.
[16] M. Mazuran, G. D. Tipaldi, L. Spinello, W. Burgard, and C. Stachniss, "A statistical measure for map consistency in slam," in *ICRA*. IEEE, 2014, pp. 3650–3655.
[17] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.
[18] S. Särkkä, *Bayesian filtering and smoothing*. Cambridge University Press, 2013, vol. 3.
[19] P. Agarwal, "Robust graph-based localization and mapping," Ph.D. dissertation, PhD thesis, University of Freiburg, Germany, 2015.
[20] S. Wang, H. Wen, R. Clark, and N. Trigoni, "Keyframe based large-scale indoor localisation using geomagnetic field and motion pattern," in *IROS*, 2016.
[21] R. Clark, S. Wang, N. T. Andrew Markham, and H. Wen, "Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization," in *CVPR*, 2017.